มาทำความรู้จักกับตัว BIOS
บริการต่าง ๆ ที่ BIOS มีให้
พารามิเตอร์ต่าง ๆ ที่ส่งผ่าน BIOS
และวิธีการเรียกใช้
บริการจาก BIOS

BIOS ของเครื่องไมโครคอมพิวเตอร์ 16 บิต

ดร.ไพศาล สงวนหมู่

ถ้าท่านเป็นผู้ใช้โปรแกรมสำเร็จรูปต่างๆ เช่น Word Star, LOTUS หรือ SideKick ท่านจะต้องเรียนรู้วิธีการใช้โปรแกรมสำเร็จรูปนั้นๆ ให้ถ่องแท้จึงจะสามารถใช้งานได้อย่างเต็มประสิทธิภาพ ท่านไม่จำเป็นจะต้องรู้ลึกซึ้งลงไปถึงระดับว่าเครื่องไมโครคอมพิวเตอร์ทำงานอย่างไร สำหรับนักพัฒนาโปรแกรมท่านก็จะต้องเรียนรู้ภาษาที่ท่านเขียน และจะต้องรู้อย่างลึกซึ้งด้วย ท่านจึงจะพัฒนาโปรแกรมที่ดีได้ เปรียบเสมือนกับว่า เราจะต้องเรียนรู้ประสิทธิภาพของเครื่องมือของเราให้ดีที่สุด เพื่อที่จะได้นำเครื่องมือไปใช้ในการผลิตผลงานที่ดีออกมาได้ ในบางครั้งการเรียนรู้ภาษาที่ใช้พัฒนาโปรแกรมอย่างเดียวยังไม่เพียงพอที่จะทำให้โปรแกรมดีที่สุดได้ อย่าลืมว่าเรากำลังพัฒนาโปรแกรมสำหรับใช้งานบนฮาร์ดแวร์ตัวหนึ่ง การเรียนรู้ฮาร์ดแวร์ที่โปรแกรมนั้นจะไปทำงานก็ย่อมจะทำให้โปรแกรมนั้นมีคุณภาพและน่าใช้ยิ่งขึ้น แทนที่จะปล่อยให้ภาษาที่ท่านใช้พัฒนาจัดการให้อย่างเดียว สำหรับนักพัฒนาระบบแน่นอนว่าท่านจะต้องเรียนรู้การทำงานของเครื่องจึงจะสามารถพัฒนาระบบได้ เพราะเครื่องมือของท่านคือ ฮาร์ดแวร์และ operating system ของฮาร์ดแวร์

ฉะนั้น ถ้าท่านเป็นนักพัฒนาโปรแกรม ถ้าท่านเป็นนักพัฒนาระบบ ท่านควรจะและจะต้องเรียนรู้การทำงานของระบบให้ดี BIOS เป็นส่วนหนึ่งของระบบเป็นส่วนเชื่อมโยงระหว่างฮาร์ดแวร์กับซอฟต์แวร์เข้าด้วยกัน จึงเป็นส่วนที่น่าจะเรียนรู้ มาซิครับเรามาทำความรู้จักกับ BIOS ของเครื่องไมโครคอมพิวเตอร์ 16 บิตกัน

BIOS คืออะไร

BIOS ย่อมาจากคำว่า Basic Input Output System ถ้าแปลตรงตัวก็คงจะได้ว่า ระบบพื้นฐานที่จัดการข้อมูลเข้าและข้อมูลออก ทำไมจึงเป็นระบบพื้นฐาน แสดงว่าจะต้องมีระบบที่สูงกว่านี้ที่จัดการเกี่ยวกับข้อมูลเข้าและข้อมูลออก ถูกแล้วครับ ระบบที่สูงไปกว่านี้ก็คือ Operating System หรือโปรแกรมจัดการของระบบ ดังนั้น BIOS จึงทำหน้าที่ประสานงานระหว่างฮาร์ดแวร์กับซอฟต์แวร์นั่นเอง

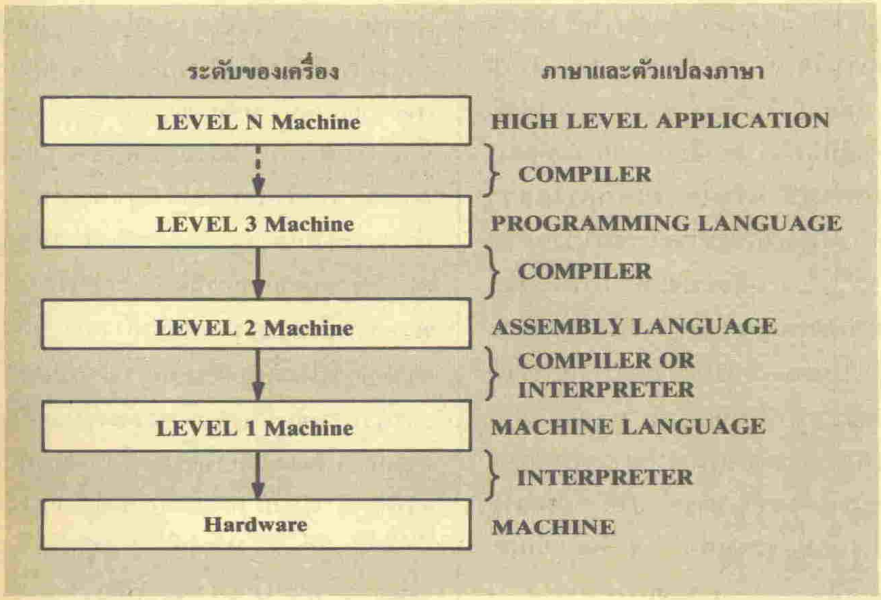
ถ้าเรามองดูฮาร์ดแวร์อย่างเดียวจะเห็นว่าเครื่องคอมพิวเตอร์ประกอบไปด้วยส่วนต่างๆ หลายส่วนที่นอกเหนือจากหน่วยประมวลผลกลาง (CPU) และหน่วยความจำ (Memory) ส่วนประกอบที่เพิ่มเติมขึ้นมา ก็เช่น คีย์บอร์ด, จอแสดงผล, หน่วยขับจานแม่เหล็ก และอื่นๆ อีก ซึ่งเรียกว่าอุปกรณ์รอบนอก (peripheral) อุปกรณ์รอบนอกเหล่านี้จำเป็นจะต้องมีโปรแกรมหรือคำสั่งควบคุมสำหรับการรับและส่งข้อมูลกับหน่วยประมวลผลกลาง เนื่องจากการควบคุมอุปกรณ์รอบนอกเหล่านี้ขึ้นอยู่กับฮาร์ดแวร์ทั้งของหน่วยประมวลผลกลางเอง และฮาร์ดแวร์ประกอบเป็นอุปกรณ์รอบนอก การควบคุมขั้นพื้นฐานจึงจะต้องขึ้นอยู่กับฮาร์ดแวร์และการควบคุมจะต้องติดมากับฮาร์ดแวร์ ดังนั้น BIOS จึงเป็นโปรแกรมที่บรรจุอยู่ใน ROM หรือ PROM เสียเป็นส่วนใหญ่ เมื่อ Operating System (หรือโปรแกรมจัดการระบบ) ต้องการ

ติดต่อกับอุปกรณ์รอบนอกจะเรียกใช้ BIOS ให้ช่วยจัดการให้อีกต่อหนึ่ง

ถ้าถามว่าทำไมจะต้องมี BIOS และ OS แยกกันให้เป็นหลายระดับด้วยล่ะ ทำไมไม่รวมงานควบคุมทุกอย่างให้ OS จัดการเองหมดเลยล่ะ มาลองดูเหตุผลของการออกแบบคอมพิวเตอร์กันดูซิครับ

คอมพิวเตอร์หลายระดับ

ดิจิตอลคอมพิวเตอร์ในระดับของฮาร์ดแวร์นั้น รับรู้คำสั่งและข้อมูลในรูปแบบของเลขฐานสองเท่านั้น การเขียนคำสั่งจะต้องเป็นในรูปของเลขฐานสองเท่านั้น ในยุคแรกของคอมพิวเตอร์ ผู้เขียนโปรแกรมจะต้องเขียนคำสั่งในรูปของเลขฐานสองที่ว่านี้ หรือแม้กระทั่งการโยกสวิตช์เปิดปิดสำหรับการโปรแกรม จะเห็นว่าโปรแกรมที่เป็นในรูปของเลขฐานสองนี้ ยากสำหรับความเข้าใจของมนุษย์ ในยุคต่อมาของคอมพิวเตอร์ในราวปี 1947 จึงมีการแบ่งคอมพิวเตอร์ออกเป็นหลายระดับ เพื่อลดช่องว่างของความยากในการเขียนคำสั่งในภาษาเครื่อง (ที่เป็นเลขฐานสอง) ให้เข้าใกล้กับภาษามนุษย์มากที่สุด รูปแบบของคอมพิวเตอร์ดังกล่าวจึงเรียกว่า Multilevel Machine (ดูรูปที่ 1 ประกอบ) ระดับล่างสุดเป็นระดับการทำงานของฮาร์ดแวร์ ระดับที่สูงขึ้นมา มีโปรแกรมควบคุมอยู่ตัวหนึ่งทำงานทำหน้าที่รับคำสั่งในระดับนี้ และเรียกใช้เครื่องในระดับที่ต่ำลงไปอีกทีหนึ่ง ดังนั้นภาษาที่ใช้ควบคุมในระดับที่สูงขึ้นจะมีความยุ่งยากกับความเข้าใจน้อยกว่าภาษาในระดับที่ต่ำลงไป และเข้าใกล้กับภาษาที่มนุษย์ใช้กันยิ่งขึ้น ถ้าเราแบ่งคอมพิวเตอร์ออกเป็น Machine หลายชั้น โดยมีจุดประสงค์ที่จะลดความยุ่งยากของภาษาเครื่องให้สามารถเขียนเป็นภาษาที่มนุษย์เข้าใจได้ จะเห็นว่าถ้าแบ่งออกเป็นหลายชั้น ความยุ่งยากหรือความแตกต่างของภาษาในระหว่างชั้นก็น้อยลงไป แต่แน่นอนครับ ในแต่ละชั้นจะต้องเสียเวลาในขบวนการในแปลงหรือแปลเป็นภาษาในระดับต่ำลงไป เรียกว่าต้องมีค่าใช้จ่าย (ซึ่งเป็นเวลา) ในระหว่างขบวนการของแต่ละชั้น ถ้าแบ่งมากชั้นเกินไป ค่าใช้จ่าย (เวลาของการทำงาน) ก็จะยิ่งมากขึ้นตาม แม้ว่าความยุ่งยากระหว่างชั้นจะน้อยลงก็ตาม ซึ่งคอมพิวเตอร์โดยทั่วไป จะแบ่งระดับดังรูปที่ 1



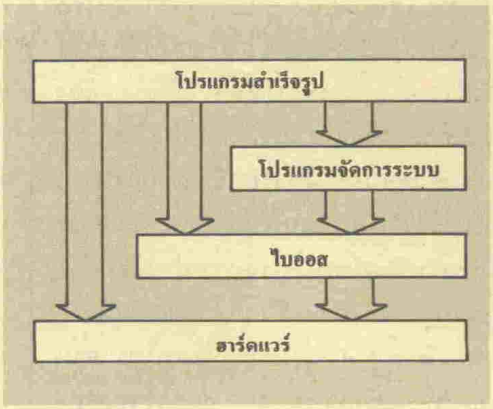
รูปที่ 1 คอมพิวเตอร์แบ่งเป็นหลายระดับ

การแปลงภาษาในระดับสูงให้เป็นภาษาในระดับต่ำนั้น ทำให้สองวิธีคือ การ Compile และ Interprete ขบวนการทั้งสองนี้แตกต่างกันก็คือ การแปล (Interprete) รับคำสั่งในระดับสูงทีละคำสั่งแล้วแปลงเป็นภาษาในระดับต่ำกว่าและทำงานทันที ส่วน compile ก็คือการแปลงคำสั่งในระดับสูงให้เป็นภาษาในระดับที่ต่ำกว่า ให้หมดทั้งโปรแกรมเสียก่อน แล้วจึงค่อยเอารหัสที่แปลงมาเสร็จเรียบร้อยไปทำงานในระดับที่ต่ำกว่าอีกทีหนึ่ง

ที่นี้เราก็จะมองเห็นคร่าวๆ ว่าทำไมเราจึงต้องแบ่งคอมพิวเตอร์ออกเป็นหลายระดับ ซึ่งพอสรุปได้ก็คือ ลดความยุ่งยากในการพัฒนาให้เครื่องสามารถรับรู้คำสั่งที่มนุษย์พอจะเข้าใจได้ หรือลดความยุ่งยากในการพัฒนาเครื่องให้เข้าใจภาษามนุษย์นั่นเอง

ระดับของเครื่องในเครื่องไมโครคอมพิวเตอร์ 16 บิต

คราวนี้ ถ้าเรามามองดูเครื่องไมโครคอมพิวเตอร์ 16 บิต จะมองเห็นเป็นระดับดังในรูปที่ 2 ซึ่งระดับต่ำสุดก็จะเป็นระดับของฮาร์ดแวร์ ระดับที่สูงขึ้นไปก็คือ BIOS ซึ่งทำหน้าที่เชื่อมโยงการทำงานระหว่าง OS และฮาร์ดแวร์ ระดับที่สูงขึ้นไป ซึ่งในที่นี้จะเรียกว่า โปรแกรมสำเร็จรูปต่างๆ



รูปที่ 2 การแบ่งระดับของเครื่องในไมโครคอมพิวเตอร์ 16 บิต

และภาษาในระดับสูง เช่น ฟอร์แทรน ปาสคาล ฯลฯ ในระดับนี้จะเรียกใช้ OS ทำงานให้ ภาษาหรือโปรแกรมสำเร็จรูปในระดับสูงที่เรียกใช้แต่ OS จะทำให้โปรแกรมหรือภาษานั้นๆ ไม่ผูกติดอยู่กับฮาร์ดแวร์นั้นคือ สามารถนำไปใช้กับเครื่องไหนก็ได้ที่ใช้ OS ตัวเดียวกัน แต่มีโปรแกรมสำเร็จรูปหลายโปรแกรมที่นอกเหนือจากการเรียกใช้ OS แล้ว ยังเรียกใช้ BIOS และเข้าควบคุมฮาร์ดแวร์โดยตรง เพื่อการทำงานที่รวดเร็วขึ้น ไม่ต้องผ่านหลายขั้นตอน โปรแกรมพวกนี้ขึ้นอยู่กับฮาร์ดแวร์ ไม่สามารถย้ายไปทำงานบนเครื่องที่มีฮาร์ดแวร์แตกต่างไปได้ ตัวอย่างเช่นโปรแกรมพวกที่จัดการแสดงผลบนหน้าจอโดยตรง เช่น LOTUS, SIDEKICK หรือ TURBO PASCAL เป็นต้น

ทำไมต้องมี BIOS ระหว่าง OS กับฮาร์ดแวร์

นอกเหนือจากการลดความยุ่งยากในการพัฒนา OS แล้ว การมี BIOS ยังสามารถทำให้ฮาร์ดแวร์สามารถเปลี่ยน OS ที่จะใช้งานได้ จะเห็นได้ว่านอกจาก MS-DOS แล้ว ยังสามารถนำ OS อื่นๆ มาใช้กับฮาร์ดแวร์ตัวเดียวกันได้ด้วย เช่น CP/M 86, UNIX และอื่นๆ นอกจากนั้นการเปลี่ยนหรือปรับปรุง OS ให้มีประสิทธิภาพยิ่งขึ้นเป็นเวอร์ชันต่างๆ สามารถทำได้ง่ายกว่าเมื่อเทียบกับไม่มี BIOS ในทำนองเดียวกันกับการปรับปรุง BIOS เพื่อให้การทำงานของเครื่องที่ดีขึ้นโดยไม่ต้องเปลี่ยนฮาร์ดแวร์ หรือเปลี่ยนฮาร์ดแวร์ (เช่น ในกรณี XT เป็น AT หรือ SYSTEM/2) ก็สามารถจะทำได้โดยไม่ต้องเปลี่ยน OS เลย

ตัวอย่างการทำงานตามลำดับชั้นของไมโครคอมพิวเตอร์

สมมติว่า โปรแกรมสำเร็จรูปเปิดแฟ้มสำหรับใช้งาน โปรแกรมหรือภาษาในระดับสูงไม่ได้ไปควบคุมฮาร์ดแวร์โดยตรงแต่จะมอบให้ OS ช่วยจัดการให้ OS ก็เรียก BIOS ต่ออีกทีหนึ่ง คำสั่งในระดับสูงอาจจะใช้คำว่า OPEN หรือ USE หรือคำสั่งที่มีชื่ออื่นใดก็แล้วแต่เมื่อ Compile หรือ Interprete ออกมาแล้วจะเป็นคำสั่งในการเรียกใช้ OS สำหรับทำหน้าที่เปิดแฟ้ม หน้าที่ของ OS ก็คือการเตรียมการสำหรับการติดต่อกับดิสก์ เป็นต้นว่า ตรวจสอบดูว่ามีชื่อแฟ้มที่จะเปิดอยู่หรือเปล่า การตรวจสอบของ OS ก็คือเรียก BIOS ให้อ่านข้อมูลในดิสเกตต์ส่วนที่เก็บไดเรคทอรีเอาไว้มาตรวจสอบดู หน้าที่ของ BIOS ก็คือ การหมุนมอเตอร์ของดิสก์ เลื่อนหัวอ่านไปตำแหน่งที่แน่นอนบนดิสก์ อ่านข้อมูลในตำแหน่งนั้น ลงในหน่วยความจำ หยุดมอเตอร์ เป็นอันหมดพิธี เป็นหน้าที่ของ DOS ที่จะต้องตรวจสอบรายชื่อแฟ้มที่อ่านออกมาเอาเอง ถ้าเจอแล้วก็ต้องจัดพิธีต่างๆ ที่จะติดต่อกับดิสก์อีก เป็นต้นว่าการสร้างพื้นที่สำหรับ FCB (หรือพื้นที่สำหรับควบคุมแฟ้ม) กำหนดหมายเลขการเข้าถึงแฟ้ม จำตำแหน่งที่อยู่ของแฟ้มบนดิสก์ เป็นต้น จะเห็นว่าภาษาหรือโปรแกรมในระดับสูงใช้คำสั่ง OPEN คำเดียว OS เอาไปทำงานหลายอย่าง พร้อมทั้งเรียกใช้ BIOS ช่วยทำงานให้ ดังนั้น BIOS จึงเปรียบเสมือนคนใช้ภายในบ้านที่จะต้องทำหน้าที่ให้แม่บ้านหรือ OS ทำงานได้สะดวกขึ้น พ่อบ้านเพียงแต่สั่งว่าเย็นนี้ชั้นอยากทานแกงไตปลานะ แม่บ้านก็ต้องมานั่งเปิดตำราดูว่าแกงไตปลามันทำอย่างไร ใช้เครื่องปรุงอะไรบ้าง จึงใช้คนใช้ออกไปหาซื้อตามท้องตลาดมาจัดการแกงให้คุณพ่อบ้านน่ะแหละ

BIOS ของไมโครคอมพิวเตอร์ 16 บิต

หลังจากอารัมภบทเรื่อง BIOS มานาน คราวนี้มาศึกษารายละเอียดของ BIOS กันเสียที BIOS ที่จะกล่าวถึงนี้เป็น BIOS ของ IBM PC เนื่องจากไมโครคอมพิวเตอร์ 16 บิตในท้องตลาดขณะนี้อยู่ในแนวทางเดียวกับ IBM PC

ภาษาระดับสูง (BASIC)	โอเปอร์เรติ้งซิสเท็ม (DOS)	ไบออส BIOS
OPEN "X.DAT" AS #1 →	INT 21H function 0FH	
	*อ่านจาก Diskette ในพื้นที่ที่เป็น Directory →	INT 13H อ่านจากข้อมูลจาก Diskette จากตำแหน่งที่แน่นอนบน Diskette ←
	*ตรวจสอบชื่อ "X.DAT"	
	*สร้างพื้นที่ควบคุมแฟ้ม (FCB)	
	*กำหนดหมายเลขแฟ้ม	
←	*รายงานผลการเปิดแฟ้ม	

รูปที่ 3
ตัวอย่างการทำงานระหว่างระดับของไมโครคอมพิวเตอร์

ทั้งนั้น ทั้งฮาร์ดแวร์ และ BIOS ก็คล้ายกันหรือเหมือนกันยังกะแกะจากเบ้าหลอมอันเดียวกันก็มี เรามาศึกษาดูว่าเราจะนำ BIOS มาใช้งานอย่างไรได้บ้าง

BIOS ทำอะไรเมื่อเริ่มเปิดเครื่อง

เมื่อเริ่มเปิดเครื่องนั้น ฮาร์ดแวร์จะเริ่มมีชีวิตชีวาได้ก็จะต้องมีโปรแกรมหรือซอฟต์แวร์เริ่มต้นการทำงานของเครื่อง นั้นคือบทบาทเริ่มต้นของ BIOS ละครับ

CPU 8088 นั้น เมื่อเริ่มต้นการทำงาน จะเรียกเอาคำสั่งแรกจากหน่วยความจำ ในตำแหน่งที่อยู่สมบูรณ์ที่ FFFF 0H (Segment FFFFH Loc. 0000H) และตำแหน่งนั้นอยู่ใน BIOS ROM ที่ตำแหน่งนั้น จะเป็นคำสั่ง JMP F000: E05B หรือกระโดดไปยังจุดเริ่มงานของ BIOS

ที่ตำแหน่ง F000:E05B BIOS เริ่มงานการตรวจสอบ ซึ่งสรุปได้ดังนี้

- *ตรวจสอบ CPU 8088 โดยการทดสอบรีจิสเตอร์ต่าง ๆ ภายใน CPU*
- *ตรวจสอบ ROM 8K แรก โดยวิธี check SUM*
- *ตรวจสอบการเรียก subroutine โดยการกำหนดจุดกลับคืน (Return address) โดยการ Push ลง Stack*

ถ้าหากมีอะไรผิดพลาดเครื่องจะหยุดทำงาน ณ จุดนี้

ที่ตำแหน่ง F000:B002 BIOS ตั้งค่าควบคุมต่างๆ สำหรับ 8237 ซัพพอร์ตชิพที่ใช้ในการควบคุมการเข้าถึงหน่วยความจำโดยตรงหรือ DMA และตรวจสอบการทำงานของวงจรเวลาใน IC 8253 รวมทั้งการตั้งค่าให้วงจรเวลาตัวแรกไปกระตุ้นให้ 8237 DMA chip ไปรีเฟรชหน่วยความจำ

ตำแหน่งที่อยู่ F000:E46A ตรวจสอบหน่วยความจำ

ตำแหน่งที่อยู่ F000:B013 เป็นการเริ่มต้น ตั้งตารางทิศทางการขัดจังหวะ (Interrupt Vector Table) สำหรับ BIOS (ดูตารางที่ 1 ประกอบ) ค่าที่นำไปใส่ในตารางจะอยู่ในหน่วยความจำของ ROM เริ่มต้นตำแหน่งที่ FEF3

ตำแหน่งที่อยู่ F000:E551 ตรวจสอบดิสเกตต์ที่มีอยู่ในระบบโดยเริ่มต้นจากแฟลกอุปกรณ์ (Equipment Flag) และตรวจสอบการทำงานของดิสเกตต์ โดยการหมุนมอเตอร์

ตำแหน่งที่อยู่ F000:E597 ตรวจสอบอะแดปเตอร์ เครื่องพิมพ์ และช่องสื่อสารแบบอนุกรม กำหนดตำแหน่งที่อยู่ของพอร์ตเหล่านั้นลงในพื้นที่การทำงานของ BIOS (ดูตารางที่ 2 ประกอบ)

ตำแหน่งที่อยู่ F000:E242 เริ่มต้น

ตารางที่ 1 ทิศทางการขัดจังหวะของ BIOS

หมายเลขการขัดจังหวะ	ตำแหน่งในหน่วยความจำ	การใช้งาน
0	F000:FF23	สำรองสำหรับกรณีหารด้วยศูนย์
1	F000:FF23	สำรองการทำงานจังหวะเดียว (สำหรับ DEBUG)
2	F000:F85F	ฮาร์ดแวร์ INT แบบ NMI (ใช้สำหรับ Parity Error)
3	F000:FF23	สำรองสำหรับจุด Break point
4	F000:FF23	สำรองสำหรับการเกิด Overflow
5	F000:FF54	สำหรับพิมพ์หน้าจอ
6	F000:FF23	สำรอง
7	F000:FF23	สำรอง
8	F000:FEA5	การขัดจังหวะตามเวลา (IRQ0)
9	F000:E987	การขัดจังหวะทางฮาร์ดแวร์ของคีย์บอร์ด (IRQ1)
A	F000:FF23	สำรองสำหรับฮาร์ดแวร์ (IRQ2)
B	F000:FF23	สำรองสำหรับช่องการสื่อสารหมายเลข 2 (IRQ3)
C	F000:FF23	สำรองสำหรับช่องการสื่อสารหมายเลข 1 (IRQ4)
D	F000:FF23	การขัดจังหวะสำหรับ Disk (IRQ5)
E	F000:FF23	การขัดจังหวะสำหรับ Diskette (IRQ6)
F	F000:FF23	สำรอง
10	F000:F065	สำหรับควบคุมการแสดงผลหน้าจอ
11	F000:F84D	ตรวจสอบอุปกรณ์ในระบบ
12	F000:F841	ตรวจสอบขนาดของหน่วยความจำ
13	F000:EC51	สำหรับการรับส่งข้อมูลระหว่าง Diskette
14	F000:E739	สำหรับการรับส่งข้อมูลของพอร์ตอนุกรม
15	F000:F859	สำหรับติดต่อกับ Cassette Tape (สำหรับ PC1)
16	F000:E82E	รับข้อมูลจากคีย์บอร์ด
17	F000:EFD2	ส่งข้อมูลออกไปที่เครื่องพิมพ์
18	F600:0000	เรียก ROM BASIC
19	F000:E6F2	บูตสแตร็ป จุดเริ่มต้นของการเรียก OS
1A	F000:FE6E	เวลาของวัน
1B	F000:FF4B	สำรองสำหรับการหยุดทำงานโดยคีย์บอร์ด
1C	F000:FF4B	สำรองสำหรับการหยุดโดยเวลา
1D	F000:F0A4	การอ่านพารามิเตอร์ของการแสดงผลบนจอ
1E	F000:EFC7	การอ่านพารามิเตอร์ของ Disk
1F	F000:0000	BIOS เพิ่มเติมสำหรับควบคุมการผลบนจอ (สำหรับ Enhance Graphic Card)

หมายเหตุ ตำแหน่งที่อยู่ในหน่วยความจำ แบ่งเป็น segment : offset

ควบคุมจอแสดงผล

ตำแหน่งที่อยู่ F000:E329 ตรวจสอบการทำงานของ IC 8259 ซึ่งทำหน้าที่สำหรับควบคุมฮาร์ดแวร์อินเตอร์รัพต์ 8 ช่อง

ตำแหน่งที่อยู่ F000:E3A2 ตรวจสอบคีย์บอร์ด

เมื่อทุกอย่างเรียบร้อย INT 19H ซึ่งเป็นตัวโหลดโปรแกรม Boot-Strap จากดิสเกตต์มาทำงาน โดยจะดึงเอาโปรแกรมที่อยู่ในดิสเกตต์ในแทร็ก 0 เซกเตอร์ 1 มาลงที่หน่วยความจำตำแหน่งที่ 0000:7C00 และมอบความรับผิดชอบทั้งปวงให้โปรแกรมที่โหลดเข้ามาใหม่ ถ้าหากการโหลดสำเร็จเรียบร้อยด้วยดี เป็นการหมดสิ้นภาระหน้าที่การเริ่มงานของ BIOS หลังจากนั้นก็คอยทำหน้าที่รับใช้เจ้านายที่เพิ่งโหลดเข้ามาละครับ หน้าที่ของโปรแกรม Boot-strap ก็คือดึงเอา OS จากดิสเกตต์มาทำงานอีกต่อหนึ่ง แต่ละ OS จะมีโปรแกรม Boot-strap loader ที่ไม่เหมือนกัน แต่จะต้องอยู่ในเซกเตอร์แรกของดิสเกตต์

ถ้าหากการดึงเอาโปรแกรม Boot-strap ไม่สำเร็จ (เพราะดิสเกตต์ใช้งานไม่ได้, เพราะดิสเกตต์ไม่พร้อมทำงาน, ฯลฯ) INT 18H จะทำงาน ซึ่งเป็นการกระโดดเข้าไปยัง BASIC ROM

BIOS ทำหน้าที่เริ่มต้นอันมากมายเหล่านี้ จึงไม่ต้องสงสัยว่า ทำไมเครื่องจึงเริ่มต้นนานเหลือเกินก่อนที่จะบูต ถ้าตัดสิ่งที่ไม่จำเป็นออกเสียบ้างก็จะบูตได้เร็วขึ้น

บริการของ BIOS

นอกจาก BIOS จะทำหน้าที่เริ่มต้นการให้ชีวิตชีวากับฮาร์ดแวร์แล้ว ยังคอยให้บริการต่างๆ เกี่ยวกับการติดต่อกับอุปกรณ์รอบนอกอีกด้วย (BIOS นี่สมกับเป็นผู้รับใช้ที่ดีจริงๆ คือสร้างเจ้านายขึ้นมาให้เรียกใช้ตัวเอง) โดยผ่านทางคำสั่งอินเตอร์รัพต์บริการต่างๆ ของ BIOS

จะเปรียบไปแล้ว BIOS ก็เหมือนคนรับใช้ภายในบ้าน ที่จะต้องทำหน้าที่ให้แม่บ้าน OS ทำงานได้สะดวกขึ้น

แสดงไว้ในตารางที่ 1 จะขอแบ่งกล่าวโดยย่อๆ ตามชนิดของการบริการดังนี้

● **INT 0H** ในกรณีที่เกิดการหารด้วยศูนย์ CPU 8088 จะกระโดดมายังตำแหน่งนี้โดยอัตโนมัติ ดังนั้นจึงถือได้ว่าตำแหน่งนี้ก็คือ TRAP สำหรับการหารด้วย 0 นั่นเอง BIOS ไม่ได้จัดการอะไรกับกรณีนี้ เพียงแต่สำรองเอาไว้ให้ OS หรือโปรแกรมอื่นจัดการเอาเอง OS หรือโปรแกรมจะต้องเปลี่ยนทิศทางการขัดจังหวะนี้ใหม่สำหรับการจัดการกับเรื่องหารด้วยศูนย์

● **INT 1H** เช่นเดียวกับ INT 0H BIOS เพียงแต่ทำ routine หลอกๆ เอาไว้ ไม่ได้จัดการอะไร ที่ตำแหน่งนี้ DOS นำไปใช้สำหรับการทำงานแบบ single step ในโปรแกรมดีบัก

● **INT 2H** เป็นตำแหน่งที่อยู่ของ routine สำหรับการขัดจังหวะทางฮาร์ดแวร์ชนิด Non Maskable ในทางฮาร์ดแวร์ การขัดจังหวะชนิดนี้จะเกิดขึ้นเมื่อการอ่านหน่วยความจำ แล้วมีบิตตรวจสอบไม่ตรงกับความจริง คงไม่ลืมว่าหน่วยความจำของ IBM-PC มีอยู่ 9 บิต บิตที่ 9 ใช้สำหรับการตรวจสอบ เมื่อวงจรฮาร์ดแวร์พบว่าบิตตรวจสอบผิดพลาดไปจากที่อ่านมาได้จากหน่วยความจำ การขัดจังหวะชนิดนี้จะเกิดขึ้น BIOS จะแสดงข้อความของการผิดพลาดและหยุดทำงาน ถ้าท่านไม่สนใจ Parity Error ท่านอาจจะเปลี่ยนทิศทางของการขัดจังหวะชนิดนี้ใหม่ได้ โดยการเปลี่ยนตารางการขัดจังหวะในหน่วยความจำที่ตำแหน่ง 000:0CH ให้ชี้ไปยัง routine ของท่าน

● **INT 3H** BIOS สำรองเอาไว้ให้โปรแกรมอื่นใช้ BIOS กำหนดเพียง routine หลอกๆ เอาไว้เท่านั้น ดอสใช้การขัดจังหวะชนิดนี้สำหรับจุดหยุดการทำงาน เมื่อกดแป้น Ctrl-Break

● **INT 4H** เช่นเดียวกับ INT 3H เก็บไว้ให้ดอสใช้สำหรับกรณี overflow

● **INT 5H** BIOS มีไว้ให้สำหรับการพิมพ์หน้าจอ ออกไปที่เครื่องพิมพ์สำหรับที่เป็นเท็กซ์เท่านั้น ไม่รวมกราฟิก เมื่อโปรแกรมควบคุมคีย์บอร์ดตรวจพบว่า Shift-Prtsc ถูกกด INT 5H จะทำงาน ถ้าท่านต้องการพิมพ์หน้าจอที่เป็นกราฟิกด้วย ท่านจะต้องเขียนโปรแกรมใหม่ควบคุมการส่งกราฟิกออกไปที่เครื่องพิมพ์เอง แล้วย้ายทิศทางของการขัดจังหวะให้ชี้ไปที่โปรแกรมของท่าน ตารางของทิศทางการขัดจังหวะนี้ อยู่ที่หน่วยความจำตำแหน่งที่ 0000:14H DOS มีโปรแกรม Graphic.com ให้ สำหรับทำหน้าที่พิมพ์กราฟิกออกที่เครื่องพิมพ์ นั่นคือ เมื่อท่านเรียก Graphic.com ออกมาทำงาน โปรแกรม Graphic.com จะเข้าไปเปลี่ยนทิศทางของการขัดจังหวะชนิดนี้เสียใหม่ให้ชี้ไปยัง routine ที่ฝังตัวเข้าไปใหม่สำหรับทำหน้าที่พิมพ์กราฟิก

● **INT 6H และ INT 7H** สำรองไว้ให้โปรแกรมอื่นใช้

● **INT 8H ถึง INT FH** สำหรับการขัดจังหวะของฮาร์ดแวร์ ซึ่งผ่านการควบคุมของ IC 8259 interrupt controller ลำดับความสำคัญของการขัด

จังหวะของฮาร์ดแวร์นี้ ไล่ไปตามลำดับตัวเลข ถ้าท่านดูสล้อตของ IBM PC จะพบว่ามีช่องทางที่จะเข้าไปใช้การขัดจังหวะของฮาร์ดแวร์เหล่านี้ผ่านทางสัญญาณ IRQ2 ถึง IRQ7 สำหรับ IRQ0 นั้น IBM ไม่ให้ใช้ เพราะต่อเอาไว้กับวงจรนาฬิกาของ IC 8253 สัญญาณ IRQ0 ถึง IRQ7 จะผ่านการกรองและจัดอันดับใน 8259 เสียก่อน และ 8259 จะเป็นผู้กำหนดทิศทางการขัดจังหวะให้ โดยการบวก 8 เข้าไปในหมายเลข IRQ (IRQ คือสัญญาณ Interrupt Request) ดังนั้น IRQ0 จะทำให้เกิดการขัดจังหวะในทิศทางที่ 8 IRQ7 ทำเกิด การขัดจังหวะในทิศทางที่ 0F H

IBM PC สำรองการใช้งานการขัดจังหวะของฮาร์ดแวร์ดังนี้

IRQ3 และ IRQ4 หมายตาเอาไว้ใช้สำหรับการสื่อสารแบบอนุกรม

IRQ5 สำหรับให้การ์ดสำหรับควบคุมฮาร์ดดิสก์ ไว้ใช้งานติดต่อกับซีพียู

IRQ6 สำหรับการ์ดสำหรับควบคุมจานแม่เหล็กอ่อน

IRQ2 และ IRQ7 สำหรับผู้ใช้นำไปประยุกต์ใช้งาน

IRQ1 สำหรับการขัดจังหวะจากคีย์บอร์ด

IRQ0 การขัดจังหวะตามสัญญาณนาฬิกา

BIOS เตรียม routine สำหรับการขัดจังหวะทางฮาร์ดแวร์ไว้เพียงสองช่องคือ IRQ0 และ IRQ1 นอกเหนือจากนั้น BIOS ไม่จัดการให้ เพียงแต่ทำ routine หลอกๆ เอาไว้ ถ้าท่านสังเกตดูในตารางที่ 1 จะพบว่ามีการขัดจังหวะหลายทิศทางที่ BIOS ตั้งค่าเอาไว้เป็น F000:FF2B นั่นคือตำแหน่งของ routine ที่ BIOS สร้างหลอกๆ เอาไว้คือใน routine ไม่ได้ทำอะไร แต่ที่ต้องกำหนดเอาไว้ก่อนเผื่อว่าเกิดมีสัญญาณขัดจังหวะทางฮาร์ดแวร์เกิดขึ้น จะได้มีที่ไป ไม่แฮงไปเสียก่อน

INT 8H เกิดขึ้นเมื่อมีสัญญาณการขัดจังหวะทางฮาร์ดแวร์ IRQ0 ซึ่งเกิดมาจากวงจรนาฬิกาใน IC 8253 ซึ่งจะส่งสัญญาณ IRQ0 ประมาณ 18.2 ครั้งใน 1 วินาที ใน routine การขัดจังหวะนี้ BIOS นำไปรักษาเวลาในเครื่องและนำไปใช้ในการควบคุมการหมุนของมอเตอร์ในหน่วยขับจานแม่เหล็กอ่อนด้วย

● **INT 9H** routine นี้จะถูกเรียกมาทำงานเมื่อมีการกดแป้นพิมพ์บนคีย์บอร์ด ซึ่งต่อสัญญาณการขัดจังหวะทางฮาร์ดแวร์เอาไว้กับ IRQ1 ทุกครั้งที่เรากดคีย์บอร์ด ไม่ว่าเครื่องจะทำอะไรอยู่ก็ตาม ถ้า CPU ยอมให้มีการขัดจังหวะเกิดขึ้น BIOS ใน routine นี้ จะรับอักขระที่กดจากคีย์บอร์ดมาเก็บเอาไว้ในหน่วยความจำ (ดูตารางที่ 2) ก่อน จำนวนสูงสุดที่ที่ BIOS สำรองเอาไว้คือ 15 อักขระ ลักษณะของข้อมูลที่รับเข้ามาเก็บไว้เป็นรหัสของแป้นพิมพ์ไม่ใช่ ASCII แป้น Shift-Prtsc และ Ctr-Alt-Del จะถูกตรวจสอบภายใน routine นี้ จะเห็นว่าที่คู่กับ Int 9H ก็คือ Int 16H ซึ่งเป็นการรับข้อมูลจากคีย์บอร์ดบัฟเฟอร์ไปใช้ในโปรแกรมนั้นคือ อักษรที่จากคีย์บอร์ดจะแบ่งเป็นสองช่วง *ช่วงแรก*เกิดการขัดจังหวะทางฮาร์ดแวร์รับรหัสของแป้นพิมพ์ไปเก็บในบัฟเฟอร์ *ช่วงที่สอง* เมื่อโปรแกรมว่างงานอย่างอื่นอาจจะถูกกำหนดให้มาตรวจสอบดูข้อมูลในคีย์บอร์ดบัฟเฟอร์เพื่อนำไปใช้งาน

บริการของ BIOS ที่มีประโยชน์สำหรับผู้ใช้จะขอนำมากล่าวตามชนิดของบริการดังนี้

บริการสำหรับควบคุมการแสดงผล (INT 10H)

การขัดจังหวะในทิศทางนี้ BIOS มีไว้สำหรับการให้บริการเกี่ยวกับการแสดงผลบนจอ โดยมีฟังก์ชันให้เลือกอีกหลายฟังก์ชัน ซึ่งกำหนดโดยค่าที่อยู่ในรีจิสเตอร์ AH

(AH) = 0 ตั้งโหมดของการแสดงผล โดยโหมดจะอยู่ใน AL

ตารางที่ 2 พื้นที่การใช้งานของ BIOS

ตำแหน่งในหน่วยความจำ	ชื่อ		การใช้งาน
40:00	RS232_BASE	DW 4DUP (?)	ตำแหน่งของพอร์ตสื่อสารอนุกรม 4 พอร์ต
40:08	PRINTER_BASE	DW 4DUP (?)	ตำแหน่งของพอร์ตเครื่องพิมพ์ 4 พอร์ต
40:10	EQUIP_FLAG	DW ?	บอกฮาร์ดแวร์ในระบบ
40:13	MEM_SIZE	DW ?	จำนวนหน่วยความจำเป็น K byte
40:17	KB_FLAG	DB ?	แฟลก สำหรับคีย์บอร์ด
40:18	KB_FLAG1	DB ?	รหัสส่วนเพิ่มของคีย์บอร์ด
40:19	ALT_INPUT	DB ?	สำหรับบอกการกดแป้น Alt
40:1A	BUFFER_HEAD	DW ?	ตัวชี้จุดเริ่มต้นของคีย์บอร์ดบัฟเฟอร์
40:1C	BUFFER_TAIL	DW ?	ตัวชี้จุดสุดท้ายของคีย์บอร์ดบัฟเฟอร์
40:1E	KB_BUFFER	DW 16DUP (?)	คีย์บอร์ดบัฟเฟอร์ 15 ที่
40:49	CRT_MODE	DB ?	สถานภาพของโหมดจอภาพ
40:4A	CRT_COLS	DW ?	จำนวนคอลัมน์บนจอภาพ
40:4C	CRT_LEN	DW ?	ความยาวหน่วยสำหรับแสดงผล
40:4E	CRT_START	DW ?	ตำแหน่งเริ่มต้นของหน่วยความจำที่แสดงผล
40:50	CURSOR_POSN	DW 8DUP (?)	ตำแหน่งของเคอร์เซอร์ใน 8 หน้า
40:60	CURSOR_MODE	DW ?	จุดเริ่มต้นของเคอร์เซอร์
40:62	ACTIVE_PAGE	DB ?	หน้าที่กำลังแสดงผลอยู่
40:63	ADDR_6845	DW ?	ตำแหน่งที่อยู่ของรีจิสเตอร์ควบคุมใน 6845

สำหรับเท็กซ์โหมด

(AL) = 0 ขาวดำ 40x25

(AL) = 1 สี 40x25

(AL) = 2 ขาวดำ 80x25

(AL) = 3 สี 80x25

สำหรับกราฟิกโหมด

(AL) = 4 สี ความละเอียด 320×200

(AL) = 5 ขาวดำ ความละเอียด 320×200

(AL) = 6 สี ความละเอียด 640×200

(AL) = 7 ขาวดำ ความละเอียด 640×200

(AH) = 1 ตั้งค่าชนิดของเคอร์เซอร์ โดยมีรีจิสเตอร์

(CH) บิตที่ 0 ถึงบิตที่ 4 บอกเส้นเริ่มต้นของเคอร์เซอร์ (CL) บิตที่ 0 ถึงบิตที่ 4 บอกเส้นสุดท้ายของเคอร์เซอร์

(AH) = 2 ตั้งค่าตำแหน่งของเคอร์เซอร์บนหน้าจอ โดยมีรีจิสเตอร์ (DH) บอกแถวนอน (DL) บอกแถวตั้ง ถ้า DH และ DL = 0, 0 เคอร์เซอร์จะอยู่มุมบนซ้ายมือ

(BH) บอกหน้าของการแสดงผลที่ต้องการเปลี่ยนตำแหน่งของเคอร์เซอร์ (สำหรับการ์ดสีจะมีหน้าสำหรับการแสดงได้ถึง 8 หน้า ในกรณีที่ใช้โหมด 1 ของการแสดงผล)

(AH) = 3 อ่านตำแหน่งปัจจุบันของเคอร์เซอร์ โดยมีรีจิสเตอร์ (BH) บอกหน้าที่ต้องการอ่าน ตำแหน่งของเคอร์เซอร์จะถูกอ่านมาในรีจิสเตอร์ (DH) สำหรับแถวนอน และ (DL) สำหรับแถวตั้ง

(AH) = 4 อ่านตำแหน่งของปากกาแสง(Light Pen)สำหรับการ์ดสีเท่านั้น เนื่องจากจอขาวดำไม่สามารถใช้ปากกาแสงได้ เพราะจอขาวดำเป็นแบบ long resistance คือสว่างอยู่ได้นาน แม้ว่าลำอิเล็กตรอนจะกวาดไปนานแล้วก็ตาม ค่าที่อ่านได้ ตอบกลับมาใน (AH)

นอกเหนือจากการลดความยุ่งยากในการพัฒนา OS แล้ว การมี BIOS ยังสามารถทำให้ฮาร์ดแวร์สามารถเปลี่ยน OS ที่จะใช้งานได้ด้วย

(AH) = 0 ลำอิเล็กตรอนของจอภาพยังกวาดไม่ถึงตำแหน่งของปากกาแสงอยู่

(AH) = 1 บอกตำแหน่งของปากกาแสงได้ ใน (DH) และ (DL) เป็นตำแหน่งของอักขระบนจอ (CH) บอกเส้นของการกวาดของลำอิเล็กตรอน (0-199) (BX) บอกจุดในแนวนอน (0-319 หรือ 0-639)

(AH) = 5 เลือกหน้าที่จะแสดงบนจอ (สำหรับโหมด 0-3 ซึ่งเป็นโหมดของการแสดงอักขระเท่านั้น)

(AL) = หน้าที่ต้องการให้แสดง (0-7 สำหรับโหมด 0 และ 1, 0-3 สำหรับโหมด 2 และ 3)

(AH) = 6 เลื่อนภาพขึ้น โดยสามารถกำหนดความกว้างบนหน้าจอของการเลื่อนขึ้นได้ในรีจิสเตอร์ (CX) และ (DX) โดย

(CH, CL) คือ แถวนอนและแถวตั้งสำหรับมุมบนซ้าย

(DH, DL) คือ แถวนอน และแถวตั้งของมุมขวาล่าง

(AL) คือจำนวนบรรทัดข้างล่างที่ต้องการให้ว่าง

ถ้าเรากำหนดให้ (CH, CL) น้อยกว่าหรือเท่ากับ (DH, DL)จะทำให้เกิดการลบทั้งจอ (clear screen)

(AH) = 7 เลื่อนภาพลง ข้อมูลที่ป้อนก็เช่นเดียวกันกับเลื่อนภาพขึ้น

(AH) = 8 อ่านอักขระและลักษณะเฉพาะ ในตำแหน่งที่เคอร์เซอร์ชี้อยู่ โดยมี (BH) เป็นหน้าของการแสดงผลที่ต้องการ อักขระที่อ่านได้อยู่ใน (AL) ลักษณะเฉพาะอยู่ใน (AH) (สำหรับเท็กซ์โหมดเท่านั้น)

(AH) = 9 เขียนอักขระและลักษณะเฉพาะลงในตำแหน่งที่เคอร์เซอร์ชี้อยู่ โดยมี

(BH) เป็นหน้าของการแสดง

(CX) จำนวนครั้งที่จะเขียนอักขระ

(AL) อักขระที่จะเขียน

(BL) ลักษณะเฉพาะ (ซึ่งเป็นตัวกำหนดสี) ที่จะเขียนไปพร้อมกับอักขระ

หลังจากเขียนแล้วเคอร์เซอร์ไม่เปลี่ยนแปลงตำแหน่ง

AH = 10 เขียนอักขระเท่านั้นลงในตำแหน่งของเคอร์เซอร์

(BH) คือหน้าที่จะเขียน

(CX) จำนวนครั้งที่จะเขียนอักขระ

(AL) คืออักขระที่จะเขียน

(AH) = 11 สำหรับในโหมดของกราฟิก เป็นการตั้งตัวกำหนดสี

(BH) รหัสประจำสี (0-127)

(BL) ค่าของสีที่ใช้กับรหัสประจำสี

(AH) = 12 เป็นการเขียนจุดลงบนโหมดของกราฟิก

(DX) = แถวนอน (0-199)

(CX) = แถวตั้ง (0-319 หรือ 0-639)

(AL) = ค่าของสี ถ้าบิตที่ 7 เป็น 1 จุดจะ Exclusive OR กับค่าที่อยู่ในตำแหน่งเดิม

(AH) = 13 อ่านค่าของจุดในตำแหน่งของรูปกราฟิกที่จุด

(DX) = แถวนอน (0-199)

(CX) = แถวตั้ง (0-319 หรือ 0-639)

(AL) จะเป็นค่าที่อ่านออกมา

(AH) = 14 เป็นการเขียนอักขระลงบนจอแบบโทรพิมพ์ในตำแหน่งที่เคอร์เซอร์อยู่ หลังจากพิมพ์อักขระไปแล้วเคอร์เซอร์จะเลื่อนไปทางขวา 1 ตำแหน่ง โดยมี

(AL) = อักขระที่จะเขียน

(BL) = สีของ foreground ในกรณีที่เป็นกราฟิกโหมด

(BH) = หน้าของการแสดงผล

(AH) = 15 อ่านค่าสถานะภาพของการแสดงผล ค่าที่อ่านออกมาได้อยู่ใน

AL เป็นโหมดของการแสดงผล

AH จำนวนแถวตั้งที่แสดง

BH หน้าที่กำลังแสดงอยู่

INT 10H ทุกฟังก์ชัน ค่าของรีจิสเตอร์ CS, SS, DS, ES, BX, CX, DX จะยังคงรักษาค่าเดิม ยกเว้นในกรณีที่อ่านค่าตอบกลับมา ค่าในรีจิสเตอร์นอกเหนือจากนี้จะถูกทำลาย

บริการสำหรับการติดต่อกับ DISKETTE (INT 13H)

BIOS มีบริการสำหรับการอ่านและเขียนดิสเกตต์คือ INT 13H นี้ โดยการอ่านหรือเขียนต้องกำหนดหมายเลขไดรฟ์ หมายเลขแทร็ก หมายเลขเซกเตอร์ และหน้าของดิสเกตต์ที่จะอ่าน เรียกว่าการเข้าถึงดิสเกตต์แบบสัมบูรณ์ (Absolute) ซึ่งแตกต่างไปจากดอสซึ่งเข้าถึงดิสเกตต์แบบสัมพัทธ์ (Relative) ซึ่งความจริงดอสก็เรียกใช้ INT 13H ในการติดต่อกับดิสเกตต์ แต่ดอสเปลี่ยนหมายเลขเซกเตอร์แบบสัมพัทธ์ให้เป็นแบบสัมบูรณ์ เพื่อที่จะให้ BIOS ทำงานได้ การที่ดอสใช้เซกเตอร์แบบสัมพัทธ์ ก็เพื่อความสะดวกในการอ้างอิงโดยใช้เลขหมายอ้างอิงเพียงตัวเดียวแทนที่จะเป็นสาม

BIOS ใช้รีจิสเตอร์สำหรับกำหนดพารามิเตอร์ต่าง ๆ ของบริการชนิดนี้ ดังนี้

(DL) หมายเลขไดรฟ์ 0 คือ A: ถึง 3 คือ D:

ค่าที่ส่งผ่านเข้าไป จะถูกตรวจสอบความถูกต้อง

เมื่อเริ่มเปิดเครื่องนั้น ฮาร์ดแวร์จะเริ่มมีชีวิตชีวาได้ ก็ต้องอาศัยซอฟต์แวร์เริ่มต้นการทำงานของเครื่อง หรือ BIOS

(DH) หัวหรือหน้าของดิสเกตต์ที่จะอ่าน (0 หรือ 1)

(CH) หมายเลขแทร็ก 0- 39

(CL) หมายเลขเซกเตอร์ 1- 8

(AL) จำนวนเซกเตอร์ที่จะอ่านหรือเขียน หรือสอบทาน

(ES:BX) ตำแหน่งของบัฟเฟอร์ที่ใช้ติดต่อกับดิสเกตต์

(AH) คือ ฟังก์ชันของการติดต่อกับดิสก์ ซึ่งมีดังนี้

(AH) = 0 เป็นการตั้งต้นใหม่ (reset) สำหรับดิสเกตต์

(AH) = 1 อ่านสถานภาพของระบบดิสเกตต์หลังจากการทำงานของดิสเกตต์ครั้งหลังสุด ผลการอ่านอยู่ใน AL

(AH) = 2 อ่านจากดิสเกตต์ลงในหน่วยความจำ

(AH) = 3 เขียนลงบนดิสเกตต์จากหน่วยความจำ

(AH) = 4 สอบทาน (Verify) เซกเตอร์ที่ต้องการ

(AH) = 5 Format แทร็กที่ต้องการ

สำหรับการ Format นั้น ES:BX จะต้องชี้ไปยังข้อมูลสำหรับการอ้างอิงในดิสเกตต์ (สำหรับเซกเตอร์นั้น ๆ) อันได้แก่ C หมายเลขแทร็ก N หมายเลขหัวเขียนอ่าน R หมายเลขเซกเตอร์และ N จำนวนไบต์ต่อเซกเตอร์ 128, 256, 512 หรือ 1024 ไบต์

ผลที่ได้จากบริการชนิดนี้ ตอบกลับมาดังนี้

(AH) สถานภาพของการทำงาน

= 0 ถ้าหากทำงานได้สำเร็จ

CY = 1 ถ้าการทำงานล้มเหลว

ถ้ามีข้อผิดพลาดเกิดขึ้นจากการทำงาน ระบบดิสเกตต์จะต้องถูกรีเซตใหม่ก่อนที่จะใช้ทำงานต่อไป

บริการสำหรับควบคุมพอร์ตอนุกรม (INT 14H)

สำหรับการควบคุมพอร์ตอนุกรม BIOS มีบริการไว้ให้ดังนี้

(AH = 0 ตั้งค่าพารามิเตอร์ของพอร์ตอนุกรม โดยมี DX บอกว่าเป็นพอร์ตไหน (0 และ 1 เท่านั้น)

AL มีค่าของพารามิเตอร์ที่จะตั้งซึ่งเข้ารหัสในรูปของบิตดังรูปที่ 4

(AH) = 1 ส่งอักขระใน AL ออกไปที่พอร์ตอนุกรม

(AH) = 2 รับอักขระจากพอร์ตอนุกรมมาใน AL

(AH) = 3 อ่านสถานภาพของพอร์ตอนุกรม

การควบคุมพอร์ตอนุกรมของ BIOS ไม่ค่อยจะเป็นที่นิยมใช้งานกัน สำหรับโปรแกรมที่ใช้ในการควบคุมการสื่อสาร นอกจากภาษาในระดับสูงเรียกให้ดอสจัดการให้ หรือคำสั่งของดอสเอง เช่น MODE, และการเปลี่ยนการรับข้อมูลจากคีย์บอร์ดเป็น COM: เช่นCTTY เป็นต้น

สำหรับโปรแกรมที่ต้องการความเร็วสูงในการจัดการกับพอร์ตอนุกรมจะเข้าไปจัดการควบคุม ฮาร์ดแวร์ (คือ 8250) โดยตรง เพราะสามารถทำได้ง่ายและเร็วกว่าการเรียกใช้ BIOS ความเร็วของการรับ/ส่ง ไม่ได้ถูกจำกัดไว้ที่ 9600 baud ตามที่ BIOS มีบริการไว้ให้ เราอาจจะเข้าไปควบคุม 8250 เองให้ส่งได้

เร็วกว่านั้นถึง 12 เท่าคือ 115200 baud

บริการสำหรับคีย์บอร์ด (INT 16H)

สำหรับคีย์บอร์ดนั้น BIOS มีการขัดจังหวะทางฮาร์ดแวร์คอยรับอักขระจากคีย์บอร์ดมาเก็บไว้ในบัฟเฟอร์ให้เราคอยเรียกมาใช้ต่อ โดยบริการของ BIOS ตัวนี้แหละ โดยจะแปลงรหัสอักขระเป็น ASCII ให้

(AH) = 0 อ่านอักขระจากบัฟเฟอร์ตัวต่อไป ค่าที่อ่านได้จะเป็นรหัส ASCII ในรีจิสเตอร์ AL และรหัสของคีย์อยู่ใน AH ถ้ายังไม่มีอักขระอยู่ในบัฟเฟอร์ BIOS จะรอจนกว่าแป้นพิมพ์ถูกกด

(AH) = 1 ตรวจสอบดูว่า มีอักขระพร้อมในบัฟเฟอร์หรือเปล่า ค่าที่ตอบกลับมา ถ้ามีอยู่ Zero flag (ZF) จะเป็น 0 และอ่านอักขระมาใน AL และ AH เช่นเดียวกันกับบริการอันแรกถ้าไม่มีอักขระอยู่ ZF จะเป็น 1 และไม่คอยแป้นพิมพ์

(AH = 2) อ่านสถานภาพของการยกแคร่ ซึ่งจะตอบกลับมาในรีจีสเตอร์ AL ดังรูปที่ 5

บริการสำหรับการควบคุมเครื่องพิมพ์ (INT 17H)

สำหรับการควบคุมพอร์ตขนานที่ต่อไว้ใช้สำหรับเครื่องพิมพ์ BIOS มีบริการให้ใช้ดังนี้

(AH) = 0 ส่งอักขระใน AL ออกไปพิมพ์

(AH) = 1 เป็นการเริ่มต้นเครื่องพิมพ์ใหม่

(AH) = 2 อ่านสถานภาพของเครื่องพิมพ์ โดยค่าที่ตอบกลับมาอยู่ใน AH ดังรูปที่ 6

ทั้งนี้ DX จะเป็นตัวกำหนดว่าใช้เครื่องพิมพ์พอร์ตไหนใน 4 พอร์ต

บริการอื่น ๆ

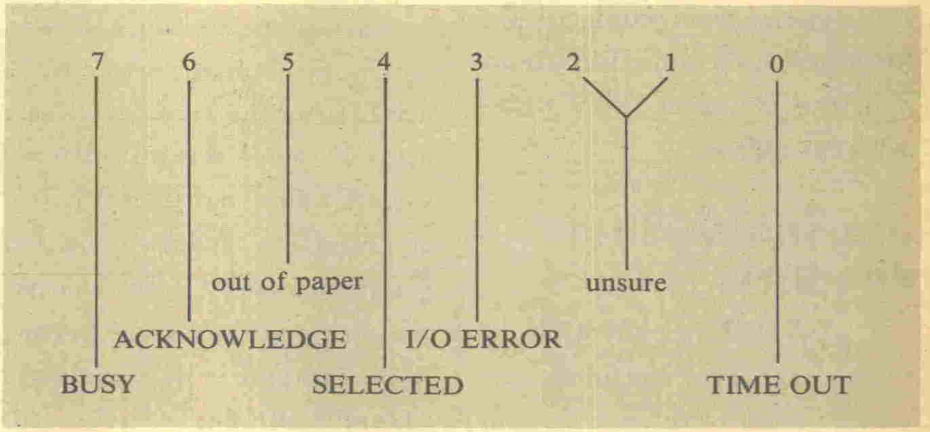
บริการอื่น ๆ ของ BIOS ดังปรากฏอยู่ในตารางที่ 1 ซึ่งไม่สู้จะมีความสำคัญ

7 6 5	4 3	2	1 0
ความเร็ว baud	การตรวจสอบ	บิตหยุด	ความยาวของคำ
000-150	Parity	0-1	10-7 บิต
001-150	×0 - ไม่ตรวจสอบ	1-2	11-8 บิต
010-300	01 - ตรวจสอบแบบคี่		
011-600	11 - ตรวจสอบแบบคู่		
100-1200			
101-2400			
110-4800			
111-9600			

รูปที่ 4 ความหมายในบิตต่าง ๆ ของ AL สำหรับการตั้งค่าพอร์ตอนุกรม

บิตที่	ความหมาย
7	1 หมายถึงกำลัง Insert อยู่
6	1 หมายถึงกำลัง Cap Lock อยู่
5	1 หมายถึงกำลัง Num Lock อยู่
4	1 หมายถึงกำลัง Scroll Lock
3	1 หมายถึง Alt-Shift กำลังกดอยู่
2	1 หมายถึง Ctrl-Shift กำลังกดอยู่
1	1 หมายถึง Shift ด้านซ้าย กำลังกดอยู่
0	1 หมายถึง Shift ด้านขวา กำลังกดอยู่

รูปที่ 5 ค่าสถานภาพของแป้นพิมพ์ เมื่อเรียกโดย INT 16H



รูปที่ 6 สถานภาพของเครื่องพิมพ์ เมื่ออ่านโดย INT 17H

เท่าใดนัก จะของดกล่าวในที่นี้ สำหรับ INT 1FH นั้น เอาไว้สำหรับการขยาย BIOS ออกไปในการควบคุมการ์ดแสดงผลที่มีความละเอียดสูง (EGA) ซึ่งจะมีมาให้ในเครื่อง IBM AT ซึ่งก็จะของดกล่าวถึงเหมือนกัน

รุ่นและลักษณะประจำตัวของ BIOS

BIOS มีการปรับปรุงมาแล้วหลายรุ่น เราสามารถจะตรวจดูรุ่นของ BIOS ได้ ในส่วนท้ายของ ROM BIOS

ตำแหน่งที่ F000 : FFF5 ถึง F000 : FFFC บอกวันที่ออก BIOS

04/24/81 PC รุ่นแรก
10/19/81 PC รุ่นปรับปรุง
08/16/82 XT รุ่นแรก
10/27/82 ยกระดับ PC ให้เท่ากับ XT
11/08/82 PC ฉบับหิ้วได้
06/01/83 PC Jr
01/10/84 AT รุ่นแรก

ตำแหน่งที่ F000 : FFFE บอกว่าตัวเองเป็นเครื่องประเภทไหน

FF PC รุ่นแรก
FE XT หรือ PC ฉบับกระเป๋า
FD PC Jr
FC AT

ลอง Debug ดูที่หน่วยความจำเหล่านี้ซิครับ ท่านจะได้รู้ว่าเครื่องของท่านอยู่ในรุ่นไหน

ตำแหน่งที่อยู่ของรูทีนต่าง ๆ ภายใน BIOS อาจจะเปลี่ยนแปลงไปบ้างในแต่ละรุ่น สำหรับที่นำมากล่าวในที่นี้ มาจากเครื่อง XT รุ่นปี 82

การเรียกใช้บริการของ BIOS

การเรียกใช้บริการของ BIOS ในภาษาระดับสูงไม่สามารถทำได้เลย นอกจากจะต้องเชื่อมโยงภาษาในระดับสูงกับภาษาระดับเครื่อง (Machine Language) เสียก่อน แล้วให้ภาษาในระดับเครื่องเรียกใช้บริการของ BIOS อีกทีหนึ่ง อย่างไรก็ตามภาษาในระดับสูงอาจจะตรวจดูผลงานของ BIOS ที่ทำเอาไว้ เป็นต้นว่า

- ตรวจดูขนาดของหน่วยความจำ
- ตรวจดู PERIPHERAL ที่ต่อไว้กับระบบ
- ตรวจดูตำแหน่งที่อยู่ของพอร์ตที่จะติดต่อกับอุปกรณ์ต่าง ๆ
- อื่น ๆ

จริงอยู่แม้ว่าผู้เขียนภาษาในระดับสูงไม่จำเป็นจะต้องรู้ระบบของเครื่อง หรือการให้บริการของ BIOS เลย แต่การเรียนรู้ระบบและพยายามเรียนรู้วิธีการเรียกใช้บริการจาก BIOS และระบบจะทำให้การเขียนโปรแกรมมีประสิทธิภาพยิ่งขึ้น ยกตัวอย่างเช่น การเขียนโปรแกรมในภาษาเบสิก เราไม่รู้เลยว่าขณะใดขณะหนึ่งเครื่องพิมพ์พร้อมที่จะทำงานหรือเปล่า ในกรณีเช่นนี้ หากสั่งพิมพ์ออกไปจะทำให้ OS ส่งข้อผิดพลาดออกมาแสดงบนจอ และโปรแกรมยุติการทำงาน ซึ่งไม่เป็นที่ต้องการของผู้ใช้ หากเราสามารถตรวจสอบเครื่องพิมพ์และเตือนผู้ใช้ก่อนล่วงหน้า ก็จะทำให้โปรแกรมนั้นน่าใช้ยิ่งขึ้น แต่การที่จะทำอย่างนั้นได้จำเป็นต้องเรียนรู้ว่าระบบมีการติดต่อกันอย่างไร เพื่อที่จะให้โปรแกรมของเราเข้าไปรับรู้สถานภาพของระบบนั้น เช่น ในกรณีเครื่องพิมพ์ที่กล่าวข้างต้น เราอาจจะตรวจสอบดูเสียก่อนว่ามีเครื่องพิมพ์ต่ออยู่หรือเปล่า โดยดูผลงานของ BIOS ที่ทำไว้ในหน่วยความจำในตำแหน่งสัมบูรณ์ที่ 408H ซึ่งจะเก็บหมายเลขพอร์ตสำหรับเครื่องพิมพ์เอาไว้หากมีเครื่องพิมพ์ต่ออยู่ ขั้นต่อไป เราก็จะต้องรู้ว่าการติดต่อกับเครื่องพิมพ์ทำกันอย่างไร เครื่องพิมพ์รายงานสถานภาพของตัวเองให้คอมพิวเตอร์รู้ที่พอร์ตหมายเลขอะไร และสถานภาพนั้นมีอะไรบ้าง ในกรณีนี้หมายเลขพอร์ตสำหรับเครื่องพิมพ์บอกสถานภาพตัวเองให้คอมพิวเตอร์รู้ จะอยู่ที่หมายเลขพอร์ตพื้นฐานบวก 1 การอ่านสถานภาพเข้ามาโดยคำสั่ง INP จะทำให้ทราบสถานภาพของเครื่องพิมพ์ดังตัวอย่างโปรแกรมภาษาเบสิกต่อไปนี้

```
10 DEF SEG = &H40 'กำหนด Segment
20 PPORT = PEEK(8) + PEEK(9)*256 'ตำแหน่งที่อยู่ของเครื่องพิมพ์ตัวแรก
30 IF PPORT = 0 THEN PRINT "No Printer Attached" : END
40 STATUS = INP (PPORT + 1)
50 IF STATUS > = 128 THEN PRINT "PRINTER NOT READY"
```

การเรียกใช้บริการของ BIOS ในภาษาเครื่องนั้น ทำได้ง่ายมาก เพียงแต่ตั้งค่าพารามิเตอร์ต่าง ๆ ให้ตรงตามความต้องการของ BIOS แล้วออกคำสั่ง INT หมายเลขบริการที่ BIOS มีไว้ให้ (ดูตารางที่ 1)

การเชื่อมต่อระหว่างภาษา แอสเซมบลี กับภาษาในระดับสูงจึงจำเป็นถ้าหากต้องการเข้าไปควบคุมระบบเอง ภาษาเบสิกไม่เหมาะสำหรับการเรียกแอสเซมบลี แม้ว่าจะทำได้แต่ค่อนข้างจะยุ่งยาก สำหรับโปรแกรมแอสเซมบลีที่ยาว ๆ เท่าที่ผมพบมาภาษา C จะเป็นภาษาที่ง่ายที่สุดในการติดต่อกับแอสเซมบลี เทอร์โบปาสคาลก็ยังนับว่าค่อนข้างจะลำบากพอสมควรเมื่อเทียบกับ C ลองหาหนังสือเทคโนโลยีไมโครคอมพิวเตอร์ 16 บิตของซีเอ็ดมาอ่านดูซิครับ จะเข้าใจการเชื่อมต่อระหว่างแอสเซมบลีกับภาษาในระดับสูงขึ้น สำหรับตัวอย่างและวิธีการนำ BIOS ไปใช้งานอย่างละเอียด คงจะต้องค้างเอาไว้เป็นโอกาสหน้าละครับ